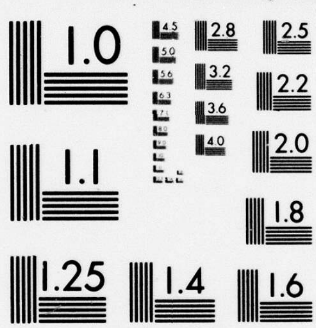
AD-A061 509 STANFORD UNIV CALIF DEPT OF OPERATIONS RESEARCH F/G 12/1
EFFICIENT HEURISTIC ALGORITHMS FOR POSITIVE 0-1 POLYNOMIAL PROG--ETC(U)
AUG 78 F GRANOT N00014-76-C-0418
UNCLASSIFIED TR-81 NL

| OF |
AD
A061509

END
DATE
FILMED
1 -79
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL II

9

6

EFFICIENT HEURISTIC ALGORITHMS FOR POSITIVE

0-1 POLYNOMIAL PROGRAMMING PROBLEMS

BY

10 FRIEDA GRANOT

9 TECHNICAL REPORT, NO. 81

AUGUST 1978

14 TR-81

11 Aug 78

PREPARED UNDER CONTRACT

15 N00014-76-C-0418 (NR-047-061)

FOR THE OFFICE OF NAVAL RESEARCH

Frederick S. Hillier, Project Director

12 29 p.

Reproduction in Whole or in Part is Permitted
for any Purpose of the United States Government

This document has been approved for public release
and sale; its distribution is unlimited.

DEPARTMENT OF OPERATIONS RESEARCH

STANFORD UNIVERSITY

STANFORD, CALIFORNIA

ACCESSION for
NTIS White Section ☒
DDC Buff Section ☐
UNANNOUNCED ☐
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

DDC
RECEIVED
NOV 27 1978
D

78 11 16 020
402 766

ABSTRACT

We consider in this paper the positive 0-1 polynomial programming (PP) problem of finding a 0-1 n-vector x that maximizes $c^T x$ subject to $f(x) \leq b$ where $c, b \geq 0$ and f is an m-vector of polynomials with non-negative coefficients.

Two types of heuristic methods for solving PP problems were developed. The various algorithms were tested on randomly generated problems of up to 1000 variables and 200 constraints. Their performance in terms of computational time and effectiveness was investiaged. The results were extremely encouraging. Optimal solutions were consistently obtained by some of the heuristic methods in over 50% of the problems solved. The effectiveness was on the average better than 99% and no less than 96.5%. The computational time using the heuristic for PP problems is on the average 5% of the time required to solve the problems to optimality.

1. Introduction

We consider in this paper the positive 0-1 polynomial programming (PP) problem of finding a 0-1 n-vector x that maximizes $c^T x$ subject to $f(x) \leq b$ where $c, b \geq 0$ and f is an m-vector of polynomials with non-negative coefficients.

Two main approaches have been proposed in the literature for solving PP problems. The first one is a linearization method which converts the PP problem to an equivalent linear 0-1 programming problem with additional variables and constraints, see e.g. [3, 4, 13]. The second approach involves solving the PP problem in its original form. Methods that can be cast into this form are Branch and Bound [7], Implicit Enumeration [10] and Covering and Generalized Covering Relaxation Algorithms [5, 6].

However, as in linear integer programming problems, all the algorithms for solving PP problems suffer from significant computational limitations. Any of those algorithms can solve only modest size problems. Motivated by this limitation and in view of the many successful heuristic algorithms that were developed for linear integer programs, see e.g. [1, 2, 8, 9, 11, 12], we will construct in this paper some heuristic methods for PP problems.

The methods we develop can be divided into two categories. The first one is a dual approach that starts by setting all variables equal to one and decreases their values, one at a time, from one to zero until feasibility is reached (see also [11, 12] for the linear case). The second approach starts with a feasible solution to PP and improves it by increasing the value of the variables until no further improvement is possible (see [9] for the linear case).

The various algorithms were tested on hundreds of randomly generated PP problems of up to 1000 variables and 200 constraints. Their performance in terms of computation time and, in problems with up to 50 variables and constraints, effectiveness, i.e., the percent the objective function value of the heuristic solution was of the optimal one, was investigated. The computational results obtained are extremely encouraging. Optimal solutions were consistently obtained by some of the heuristic methods in over 50% of the problems solved. Further, the effectiveness was on the average better than 99% and no less than 96.5%. The computational time using the heuristic for PP problems is on the average 5% of that using the covering relaxation approach described in [6].

2. Preliminary Definitions and Notations

Consider again the positive 0-1 polynomial programming PP problem

$$\text{Maximize} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{Subject to} \quad f_i(x) \le b_i \qquad i=1,\dots,m$$

$$x_j \in \{0,1\} \qquad j=1,\dots,n$$

where $f_i(x)$ are polynomials of the form

$$f_i(x) = \sum_{k=1}^{P_i} a_{ik} \prod_{j \in N_{ik}} x_j$$

with N_{ik} any subset of $N = \{1,\ldots,n\}$. We will assume that the c_j and b_i are all nonnegative and that the a_{ij} are positive.

If we further assume, without loss of generality, that $b_i > 0$ $(i=1,\ldots,m)$ (since if not, $x = 0$ is the only feasible solution), then we can normalize each polynomial function by dividing all of its coefficients by b_i. Thus we can assume without loss of generality that $b = 1$. Call this the normalized problem.

Now, for any given variable x_j, $f_i(x)$ can be written as

$$f_i(x) = x_j g_i^j(x^j) + h_i^j(x^j)$$

where $x^j = (x_1,\ldots,x_{j-1},x_{j+1},\ldots,x_n)$. We will refer to $g_i^j(x^j)$ as the derivative of $f_i(x)$ with respect to the variable x_j. Observe that if a variable x_j is increased from zero to one then $g_i^j(x^j)$ is the change in the left hand side of the i^{th} constraint. The $g_i^j(0)$ can be considered as the consumption of the right hand side incurred by such an increase. We will refer to $g_i^j(0)$ as the linearity of x_j in $f_i(x)$ and denote it by L_i^j.

We will now briefly motivate the ideas underlying our heuristic methods. When solving a PP problem we would like to first set a variable x_j to one if its contribution to the objective function is as large as possible, while its consumption of the right hand side is as small as possible. Thus we would choose to set x_j to one if $k=j$ maximized

(1) $$\frac{c_k}{\sum_{i=1}^{m} L_i^k} .$$

Further since when x_j is increased from zero to one the change in the left hand side of the i^{th} constraint is $g_i^j(x^j)$, we would like $g_i^j(x^j)$ to involve terms with small coefficients and each having a large number of variables. The main reason for that is to ensure that a further increase of other variables from zero to one, in particular those which appear in $g_i^j(x^j)$, will consume only a small amount of the right hand side. This leads to the definition of the weighted linearity of x_j in $f_i(x)$, denoted by WL_i^j, and given by

$$(2) \qquad WL_i^j = \sum_{k \in I_i^j} a_{ik}/|\bar{N}_{ik}|$$

where $|\bar{N}_{ik}|$ is the cardinality of $\bar{N}_{ik}$ and $g_i^j(x^j)$ is given by

$$(3) \qquad g_i^j(x^j) = \sum_{k \in I_i^j} a_{ik} \prod_{r \in \bar{N}_{ik}} x_r$$

where I_i^j and $\bar{N}_{ik}$ are the appropriate sets defining g_i^j.

If on the other hand we would like to initially set all variables equal to one, the infeasibility of the i^{th} constraint will be given by $(f_i(1) -1)^+$. If x_j is then decreased from one to zero, the change in the left side will be given by $g_i^j(1)$ and the infeasibility of the i^{th} constraint will drop to $(h_i^j(1) - 1)^+$ where $f_i^j(x) = x_j g_i^j(x^j) + h_i^j(x^j)$. Now starting with $x = 1$ and attempting to solve the PP problem we would like to decrease first that variable x_j whose contribution to the objective function is as small as possible while its consumption of the right hand side is as large as possible in the violated contraints. Thus we would decrease the value of the variable x_j for which S^j , given by

(4)
$$S^j = \frac{c_j}{\sum_{i=1}^{m} (f_i(1) - 1)^+ \, g_i^j(1)} ,$$

is minimum.

Moreover if $g_i^j(x^j)$ contains terms with large coefficients and which involve a large number of variables, those terms will be discarded when x_j is set to zero. Thus we can modify (4) and choose to decrease to zero that variable for which WS^j , defined by

$$WS^j = \left\{ \frac{c_j}{\sum_{i=1}^{m} \left(\sum_{k \in I_i^j} |\bar{N}_{ik}| \, a_{ik} \right) (f_i(1) - 1)^+} \right\} ,$$

is minimized.

The quantities L_i^j , WL_i^j , S^j and WS^j are used in the next section where the heuristic algorithms for solving PP are described.

3. The Heuristic Algorithms

We will present in this section two types of heuristic algorithms for solving the PP problem. The first type will start with a feasible solution

$x = 0$ and increase the values of the variables one at a time until no further increase is possible. The order in which the variables are increased is determined using once their sum of linearities and then the sum of their weighted linearities. The second type of algorithm will start with an infeasible solution $x = 1$ and use the infeasibility as well as the change in infeasibility to determine variables to be dropped to zero. This is done until a feasible solution is reached. Once a feasible solution is at hand, an improvement procedure will be applied to determine whether some of the variables dropped to zero can be increased back to one. This procedure seems to play a crucial role in the excellent performance of the second type of algorithms on the PP problem.

The six different heuristic algorithms can now be formally stated as follows:

Algorithm I:

Step 0: Start by setting $x = 0$, $I_0 = \{1,\ldots,n\}$,

$I_F = \emptyset$ and $M = \{1,\ldots,m\}$.

Step 1: Let j be an index $k \in I_0 \setminus I_F$ maximizing

$$\frac{c_k}{\sum_{i \in M} L_i^k} \ .$$

<u>Step 2:</u> Check whether by increasing x_j from zero to one any of the constraints will be violated. If yes go to step 3; otherwise go to step 4.

<u>Step 3:</u> Set $I_F = I_F \cup \{j\}$ and $I_0 = I_0 \setminus \{j\}$.

If $I_0 = \emptyset$ go to step 5; otherwise go to step 1.

<u>Step 4:</u> Set $x_j = 1$, subtract L_i^j from both sides of the i^{th} constraint, and normalize the problem, thereby determining modified a_{ij} and L_i^j . Set $I_0 = I_0 \cup (I_F \setminus \{j\})$, $I_F = \emptyset$. If $I_0 = \emptyset$, go to step 5; otherwise if the k^{th} constraint is redundant set $M = M \setminus \{k\}$ and go to step 1.

<u>Step 5:</u> Terminate with x.

Algorithm I can be modified, using the weighted linearities, to produce

<u>Algorithm II:</u>

Same as algorithm I except replace step 1 by

<u>Step 1':</u> Let j be an index $k \in I_0 \setminus I_F$ maximizing

$$\frac{c_k}{\sum_{i \in M} WL_i^k} .$$

The next four algorithms will start with the optimal solution $x = 1$ to the relaxation of the PP problem in which the polynomial constraints are dropped and proceed towards feasibility.

<u>Algorithm III</u>

<u>Step 0:</u> Start with $x = 1$. If x is feasible for PP, terminate. Otherwise set $J = \{1,\ldots,n\}$.

<u>Step 1:</u> Determine the set I of indices i for which $f_i(x) > 1$. Go to step 2.

<u>Step 2:</u> Let k be an index j in J minimizing

$$s^j = \frac{c_j}{\sum_{i \in I} (f_i(x) - 1)^+ \; g_i^j(x^j)}$$

Set $x_k = 0$. If x is feasible, terminate with x; otherwise replace J by $J \backslash \{k\}$ and go to step 1.

Step 2 of algorithm III can be further modified resulting with algorithm IV as follows:

<u>Algorithm IV</u>

Same as algorithm III except that in step 2 we let k be an index j that minimizes

$$WS^j = \frac{c_j}{\sum_{i \in I} (f_i(x) - 1)^+ \sum_{r \in I_i^j} |\bar{N}_{ir}| \, a_{ir}}$$

where $\bar{N}_{ir}$ and I_i^j are defined in (3).

Observe that in algorithms III and IV whenever a variable x_i is set to zero the algorithms return to step 1 and calculate the next best candidate to be set to zero. The following two algorithms are similar to algorithms III and IV, however the variables are ordered once and then set to zero one at a time until feasibility is reached.

Algorithm V:

Step 0: Start with $x = 1$. If x is feasible for PP, terminate. Otherwise go to step 1.

Step 1: Let $j_1, \ldots, j_n$ be a permutation of the first n integers such that $j_i < j_k$ if $S^{j_i} < S^{j_k}$ or if $i < k$ and $S^{j_i} = S^{j_k}$. Set x_j equal to zero, in order, one at a time, until feasibility is reached. Terminate with x.

Algorithm VI:

Same as algorithm V except the variables are ordered in increasing order of WS^j.

The solution x determined by any of the algorithms III to VI can be further improved. This improvement was found to play a significant role in the effectiveness of the methods of the second type. To motivate this procedure observe that a decision to drop the value of a given variable from one to zero may very well be reversed after other variables having the value one are forced to zero. Thus after determining a solution x, we would like to check whether any of the variables dropped to zero can be increased back to one without violating feasiblity. This can be easily done using the following procedure:

<u>Modification Procedure:</u>

<u>Step 0:</u> Start with x -- the solution obtained by any of the heuristic algorithms III to VI. Let $\overline{PP}$ be the problem obtained from PP after substituting the values of all variables x_j that are equal to one in x. (Thus $\overline{PP}$ involves only those variables whose value is zero in x.)

<u>Step 1:</u> Use heuristic method II for $\overline{PP}$, terminating with $\hat{x}$. Decrease to one the values of those variables in the x from step 0 that are equal to one in $\hat{x}$, and terminate with this new x.

The above improvement process can be easily modified to incorporate cases where we would like to test whether a decrease of another variable from one to zero can improve the heuristic solution. This is done by eliminating the substitution of that variable in PP in step 0 of the modification process.

4. Computational Results

The six algorithms were coded in Fortran IV and implemented on an IBM 370/168. The performance of the algorithms was tested on a large number of randomly generated problems and was measured in terms of computational time and effectiveness, i.e., the percent the objective function value of the heuristic solution was of the optimal one. For the larger problems with up to 1000 variables and 200 constraints, for which an optimal solution was not sought, the algorithms were compared according to their relative effectiveness.

We have tested the performance of the heuristic algorithms in terms of five parameters, viz., the number n of variables, the number m of constraints, the maximum number k of terms per constraint, the maximum number v of variables in each term, and the degree of tightness α of the constraints.

The data for each PP problem was randomly generated as follows. The cost vector c was determined by setting $c_0 = 0$ and $c_{j+1} = c_j + \rho$ where ρ is randomly chosen from $[0,10]$. The coefficients a_{ij} are randomly chosen between 0 and 10, while the right hand side b_i was set equal to $\alpha \cdot \sum_{j=1}^{m} a_{ij}$, with α equal to 0.3, 0.5 or 0.9. The number of terms in each constraint was randomly chosen between 1 and k and the number of variables in each term was randomly chosen between 1 and v.

All possible combinations obtained by varying n and m among 30, 40, 50 and α among 0.3, 0.5 and 0.9 were tested. From the results obtained for the 270 problems that were run, ten for each combination of n, m and α, we can conclude that method II dominates method I, and methods III and IV are uniformly superior to methods V and VI both in terms of efficiency and in computational time. There was no clear-cut choice between methods III and IV. All problems were initially solved with the original algorithms without adding the modification procedure. The results obtained by method II were extremely good and the effectiveness was never below 96.5%. The performance of methods III and IV was not as good and in some cases, especially when the constraints were tight, the effectiveness sometimes fell to 94-95%. However, both methods III and IV required less computation time to produce their heuristic solutions. After employing the modification to methods III and IV the solutions were uniformly improved and many of them reached the optimal value. In most cases, except when the constraints were very loose, the total computational time required by the modified methods III and IV exceeded that of method II. Of the many randomly-generated problems solved, method II reached optimality in about 50% of the problems, while methods III and IV did so in about 30% of the problems. After employing the modification process on methods III and IV, they reached optimality in about 60% of the problems solved.

The influence of the number of variables, number of constraints and tightness of the constraints on computational time and effectiveness in algorithm II and the modified algorithms III and IV is summarized in Table 1 and Figure 1. The slopes a and the intercepts b for each line in Figure 1 were obtained using averages from all runs in which one of the factors was kept constant and all others were changed in their given ranges.

The effectiveness of all three methods was not affected by either the number of constraints or the number of variables. However, as shown in Figure 1 the effectiveness decreases for method II and increases for methods III and IV with a decrease in constraint tightness. Computation time, however, increases for all three methods with an increase in the number of variables and/or constraints. This increase in computation time is strongly influenced by the tightness of the constraints. Indeed method II is faster for tight constraints, while methods III and IV are faster for loose constraints. By comparing the performance of the three methods we observe that for $\alpha = 0.3$ and 0.5,

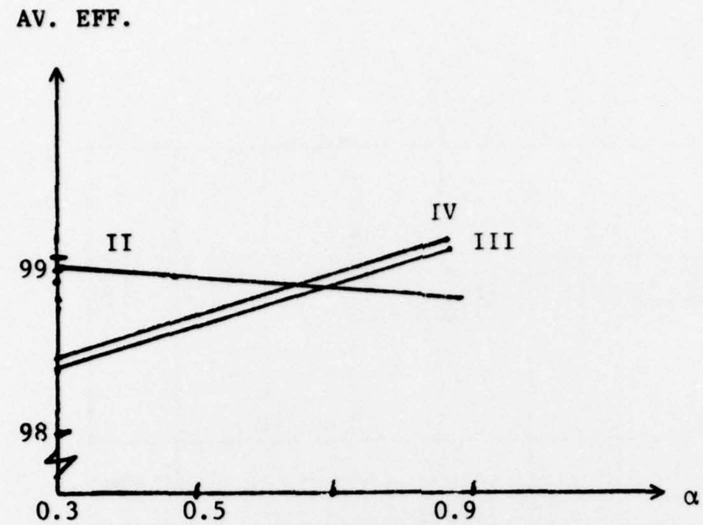
Dimensions					Method II			Modified Method III			Modified Method IV			Covering Relaxation Method
n	m	k	v	α	AV. EFF.*	TIME†	%AT**	AV. EFF.*	TIME†	%AT**	AV. EFF.*	TIME†	%AT**	TIME
30	30	5	5	0.3	97.82	24	4.49	97.68	58	10.79	97.68	60	11.01	541
30	40	5	5	0.3	98.56	27	5.73	98.87	74	15.47	98.83	79	16.43	478
30	50	5	5	0.3	98.89	30	5.22	98.75	82	14.31	98.28	87	15.11	574
40	30	5	5	0.3	97.74	40	2.46	98.38	66	4.12	98.61	67	4.17	1611
40	40	5	5	0.3	99.63	40	2.68	99.31	72	4.84	99.31	76	5.15	1485
40	50	5	5	0.3	98.99	41	0.52	98.99	87	1.10	98.99	93	1.18	7913
50	30	5	5	0.3	97.63	52	0.78	98.31	77	1.17	98.68	78	1.18	6576
50	40	5	5	0.3	99.35	54	1.29	99.32	83	1.98	98.88	86	2.04	4215
50	50	5	5	0.3	99.12	63	0.32	98.90	95	0.49	99.09	97	0.49	19634
30	30	5	5	0.5	98.22	40	2.32	99.07	63	3.65	99.58	65	3.79	1717
30	40	5	5	0.5	98.25	42	1.87	98.95	72	3.21	98.23	73	3.29	2232
30	50	5	5	0.5	98.14	55	3.54	99.02	85	5.47	98.61	87	5.59	1558
40	30	5	5	0.5	97.93	55	0.71	98.23	60	0.77	99.17	65	0.85	7777
40	40	5	5	0.5	98.45	62	0.92	98.77	69	1.02	98.98	74	1.09	6779
40	50	5	5	0.5	99.23	67	0.59	98.25	92	0.82	99.26	91	0.81	11240
50	30	5	5	0.5	97.86	79	0.46	97.63	65	0.38	97.54	73	0.42	17133
50	40	5	5	0.5	98.99	84	0.10	97.83	85	0.10	97.73	87	0.11	82250
50	50	5	5	0.5	99.65	98	0.11	98.37	99	0.11	98.22	107	0.12	86896
30	30	5	5	0.9	97.73	47	15.60	99.60	53	17.73	99.60	53	17.73	301
30	40	5	5	0.9	97.75	57	15.35	99.41	62	16.69	99.41	63	16.85	373
30	50	5	5	0.9	98.41	61	8.43	99.21	67	9.28	99.66	70	9.73	724
40	30	5	5	0.9	98.36	61	13.36	99.82	53	11.62	99.82	59	12.84	460
40	40	5	5	0.9	97.54	71	7.86	99.22	64	7.08	98.85	65	7.17	901
40	50	5	5	0.9	98.22	85	2.30	99.02	74	1.99	98.80	85	2.29	3704
50	30	5	5	0.9	97.66	95	13.35	98.38	61	8.59	98.44	62	8.73	715
50	40	5	5	0.9	98.69	100	5.39	99.05	72	3.89	99.17	72	3.88	1851
50	50	5	5	0.9	98.24	114	0.87	98.89	78	0.60	98.89	83	0.63	13061

Table 1

* Average effectiveness in the sample of ten.

** %AT equals the percentage that the average computation time required by the heuristics is of the average computation time to find an optimal solution by the Covering Relaxation Method.

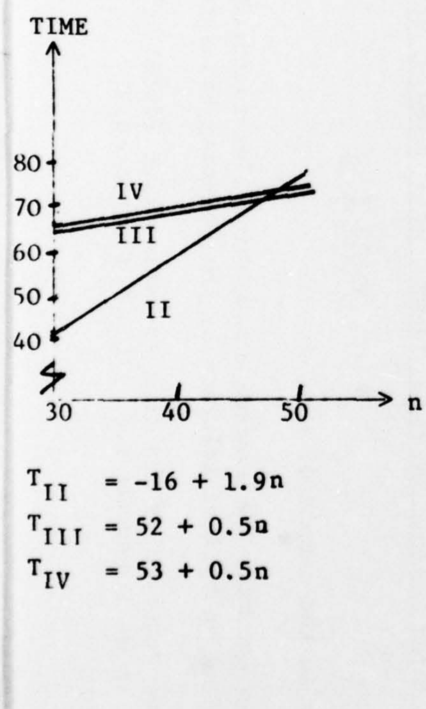
† The computational time reported is the true execution time (total CPU time minus loading input/output and overhead time) given in milliseconds i.e. 1/1000 of a second.



$AV.EFF_{II} = 99 - 0.4\alpha$

$AV.EFF_{III} = 98 + 1.2\alpha$

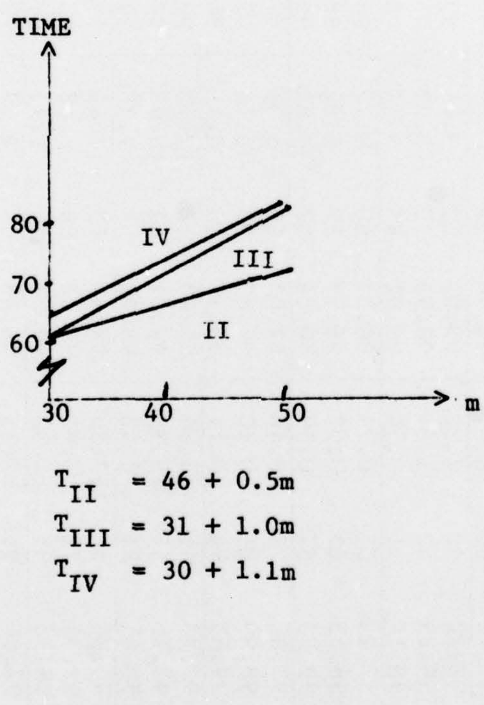
$AV.FFF_{IV} = 98 + 1.3\alpha$



$T_{II} = -16 + 1.9n$

$T_{III} = 52 + 0.5n$

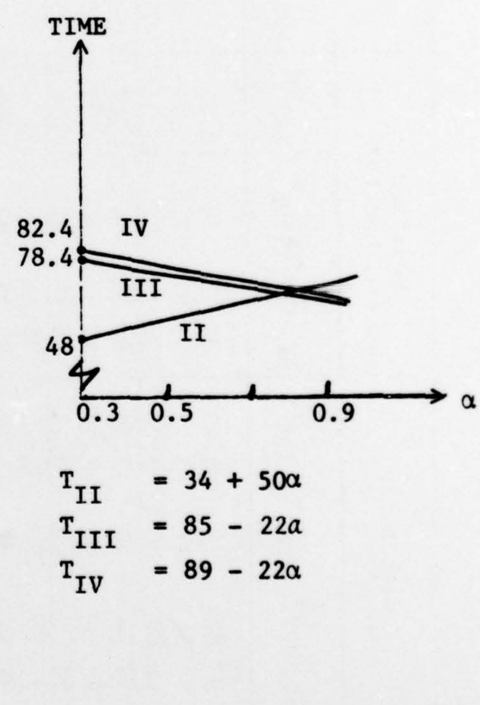
$T_{IV} = 53 + 0.5n$



$T_{II} = 46 + 0.5m$

$T_{III} = 31 + 1.0m$

$T_{IV} = 30 + 1.1m$



$T_{II} = 34 + 50\alpha$

$T_{III} = 85 - 22a$

$T_{IV} = 89 - 22\alpha$

Figure 1

method II is almost uniformly faster than methods III and IV. However, for $\alpha = 0.9$ methods III and IV become faster than method II, especially for the larger problems. In fact, for $\alpha = 0.9$ methods III and IV without the modification process are as effective as method II. Moreover, they require significantly less computation time to produce the heuristic solutions than is required by method II. These results are summarized in Table 2. Indeed in Table 2 we observe that the effectiveness of the three methods is about the same and that methods III and IV are almost three times faster than method II.

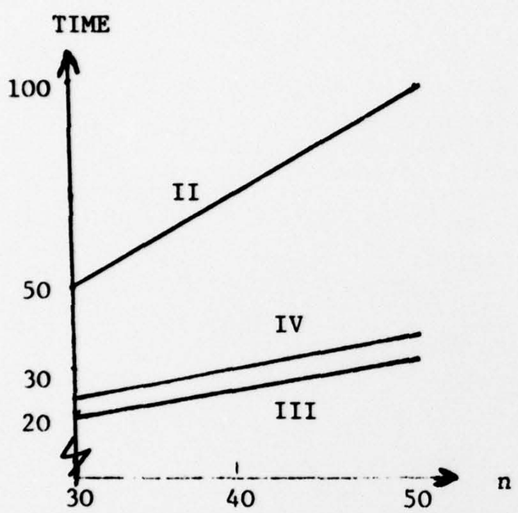
In Table 3 and Figure 3 we exhibit the influence of a change in the maximum number of terms k in each constraint and the maximum number of variables v in each term on both effectiveness and computation time.

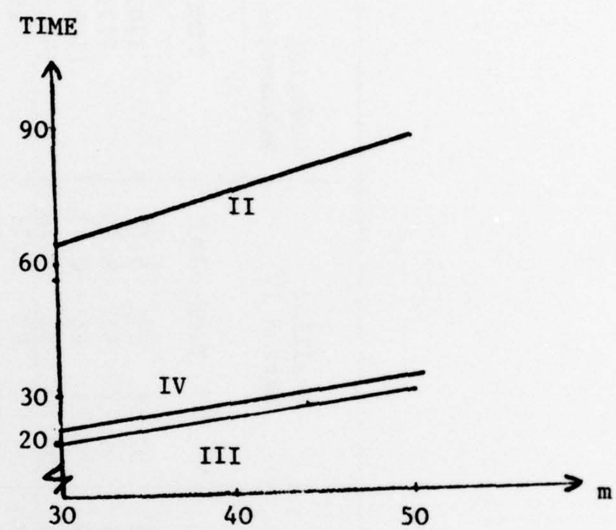
The computation time is increased for all three algorithms with an increase in either k or v, however with a much larger slope for method II than for methods III and IV. Again the effectiveness of neither of these methods is superior to the other two. It is interesting to note that when either k or v is very small (average of 2) the effectiveness of all three methods is worst.

The performance of methods II and modified methods III and IV was then tested on some large problems of up to 1000 variables and 200 constraints that were generated in a similar way to those generated before. No attempt has been made to find optimal solutions for those problems because of the excessive computation time it would require. The performance of the three methods was compared by means of their effectiveness relative to the best heuristic obtained and by time relative to the worst time obtained. The results were averaged from ten runs and are summarized in Table 4.

Dimensions					Method II			Method III			Method IV			Covering Relaxation Method
n	m	k	v	α	AV. EFF.	TIME	%AT	AV.EFF.	TIME	%AT	AV. EFF .	TIME	%AT	TIME
30	30	5	5	0.9	97.73	46	15.41	99.06	19	6.18	99.42	20	6.57	301
30	40	5	5	0.9	97.75	53	14.17	98.41	21	5.74	98.82	23	6.28	373
30	50	5	5	0.9	98.41	61	8.43	98.91	20	2.74	99.66	24	3.26	724
40	30	5	5	0.9	98.36	64	14.01	99.43	21	4.57	99.30	23	4.96	460
40	40	5	5	0.9	97.54	70	7.82	98.58	25	2.82	98.26	25	2.73	901
40	50	5	5	0.9	98.22	82	2.21	98.66	26	0.71	98.31	28	0.77	3704
50	30	5	5	0.9	97.66	85	11.87	98.03	24	3.33	97.83	30	4.23	715
50	40	5	5	0.9	98.69	105	5.69	98.87	29	1.56	98.81	32	1.72	1851
50	50	5	5	0.9	98.24	117	0.90	98.33	33	0.25	98.33	35	0.26	13061

Table 2





$T_{II} = -22.1 + 2.4n$

$T_{III} = 6.89 + 0.4n$

$T_{IV} = 6.67 + 0.5n$

$T_{II} = 32.6 + 1.1m$

$T_{III} = 14.22 + 0.2m$

$T_{IV} = 17.33 + 0.2m$

Figure 2

Dimensions					Method II			Modified Method III			Modified Method IV			Covering Relaxation Method
n	m	k	v	α	AV. EFF.	TIME	%AT	AV. EFF.	TIME	%AT	AV. EFF.	TIME	%AT	TIME
40	30	3	5	0.5	97.97	37	3.34	97.89	59	5.33	97.59	57	5.17	1107
40	30	5	5	0.5	97.93	55	0.71	98.23	60	0.77	99.17	65	0.85	7777
40	30	7	5	0.5	98.45	88	0.44	97.91	77	0.39	98.11	85	0.43	19850
40	30	9	5	0.5	99.02	98	0.27	98.35	85	0.24	98.80	91	0.25	35804
40	30	5	3	0.5	97.75	39	1.69	97.64	59	2.54	97.82	63	2.71	2319
40	30	5	5	0.5	97.93	55	0.71	98.23	60	0.77	99.17	65	0.85	7777
40	30	5	7	0.5	98.70	68	4.80	98.29	61	4.28	98.86	68	4.78	1416
40	30	5	9	0.5	98.81	86	5.16	99.60	65	3.88	99.68	75	4.49	1674

Table 3

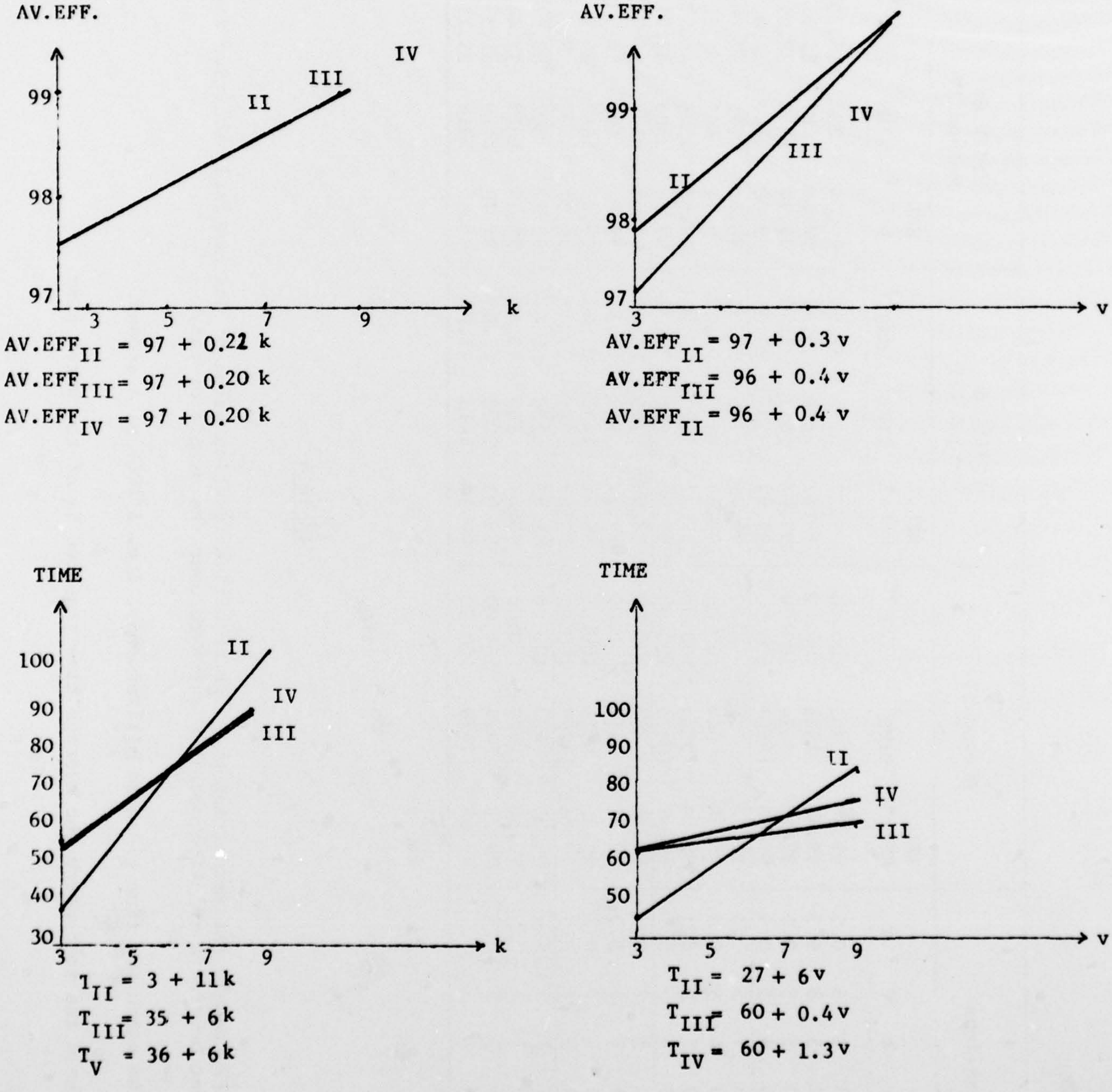

AV.EFF.
IV
III
II
99
98
97
3
5
7
9
k
$AV.EFF_{II} = 97 + 0.22\ k$
$AV.EFF_{III} = 97 + 0.20\ k$
$AV.EFF_{IV} = 97 + 0.20\ k$
AV.EFF.
99
IV
III
II
98
97
3
v
$AV.EFF_{II} = 97 + 0.3\ v$
$AV.EFF_{III} = 96 + 0.4\ v$
$AV.EFF_{II} = 96 + 0.4\ v$
TIME
100
90
80
70
60
50
40
30
II
IV
III
3
5
7
9
k
$T_{II} = 3 + 11k$
$T_{III} = 35 + 6k$
$T_{V} = 36 + 6k$
TIME
100
90
80
70
60
50
II
IV
III
3
5
7
9
v
$T_{II} = 27 + 6v$
$T_{III} = 60 + 0.4v$
$T_{IV} = 60 + 1.3v$

Figure 3

Dimensions					Method II			Modified Method III			Modified Method IV			
n	m	k	v	α	REL. EFF.*	TIME†	%AT**	REL. EFF.*	TIME†	%AT**	REL. EFF.*	TIME†	%AT**	WORST TIME
100	100	5	5	0.5	98.35	342	100.00	99.27	214	62.65	99.04	236	68.85	342
200	200	5	5	0.5	99.48	1334	100.00	99.82	612	45.88	99.68	687	51.51	1334
500	200	5	5	0.5	99.58	5614	100.00	99.87	1611	28.70	99.74	1853	33.01	5614
1000	200	5	5	0.5	99.55	13457	100.00	99.99	3131	23.35	99.80	3600	26.85	13407
100	100	5	5	0.3	99.27	195	92.67	98.99	198	94.38	98.63	195	92.86	210
200	200	5	5	0.3	98.38	643	100.00	98.85	473	73.55	99.71	491	76.32	643
500	200	5	5	0.3	99.34	3825	100.00	99.77	1453	27.99	99.82	1598	41.77	3825
1000	200	5	5	0.3	99.49	10635	100.00	99.90	3266	30.71	99.94	3563	33.51	10635
100	100	5	7	0.5	99.39	382	100.00	98.99	195	51.05	98.39	209	54.71	382
200	200	5	7	0.5	99.82	1418	100.00	99.34	589	41.53	99.07	633	44.66	1418
500	200	5	7	0.5	99.75	6621	100.00	99.85	1456	22.00	99.85	1641	24.79	6621
1000	200	5	7	0.5	99.76	16968	100.00	99.87	2955	17.42	99.87	3183	18.76	16968
100	100	7	5	0.5	99.30	472	100.00	98.83	276	58.50	98.77	302	63.93	472
200	200	7	5	0.5	99.56	1929	100.00	98.84	880	45.61	98.58	1012	52.44	1929
500	200	7	5	0.5	99.22	7977	100.00	99.93	2208	27.68	99.79	2499	31.33	7977
1000	200	7	5	0.5	99.65	19979	100.00	99.94	4414	22.09	99.82	5016	25.11	19979

Table 4

* REL. EFF. equals the percentage that the objective function value of the heuristic solution derived is of the best heuristic solution obtained averaged over the sample of ten.

† The true execution time given in milliseconds, i.e. 1/1000 of a second.

** %AT equals the percentage the average time required is of the worst time.

From Table 4 we observe that the relative effectiveness of modified methods III and IV seem to improve slightly with an increase in the number of variables. It is interesting to note that the relative effectiveness of the three methods remains nearly constant, ranging between 98.39 to 99.99%. The relative time on the other hand decreases almost linearily for modified methods III and IV. This makes them more attractive for large size problems.

ACKNOWLEDGEMENT

The author wishes to acknowledge the programming assistance of Mr. W. Vaessen. The author is also greatful to Professor Daniel Granot for valuable discussions on the subject, and to Professor A. Veinott for his many helpful remarks on an earlier draft of the paper.

References

[1] BALAS, E. and C. H. MARTIN, "Pivot and Complement - A Heuristic for 0-1 Programming", in MSR Report No. 414, Carnegie Mellon University, February 1978.

[2] FAALAND, D. H. and F. S. HILLIER, "Interior Path Methods for Heuristic Integer Programming Procedures", Technical Report #73, Department of Operations Research, Stanford University, February 1977.

[3] GLOVER, F. and E. WOOLSEY, "Further Reduction of Zero-One Polynomial Programming Problems to Zero-One Linear Programming Problems", Operations Research, Vol. 21, No. 1 (1973), pp. 141-161.

[4] GLOVER, F. and E. WOOLSEY, "Converting the 0-1 Polynomial Programming Problem to a 0-1 Linear Program", Operations Research, Vol. 22, No. 1 (1974), pp. 180-182.

[5] GRANOT, D. and F. GRANOT, "Generalized Covering Relaxation for 0-1 Programs", Technical Report SOL 78-16, Stanford University, June 1978.

[6] GRANOT, D., F. GRANOT and J. KOLLBERG, "Covering Relaxation in Monotone 0-1 Programming", Submitted to Management Science.

[7] HAMMER, P. L., "A B-B-B Method for Linear and Nonlinear Bivalent Programming", Operations Research, Statistics and Economics Mimeograph Series No. 48, Technion, May 1969.

[8] HILLIER, F. S., "Efficient Heuristic Procedures for Integer Linear Programming with an Interior", Operations Research, 17 (1969), pp. 600-637.

[9] KOCKENBERG, G. A., B. A. MCCOIL and F. P. WYMAN, "A Heuristic for General Integer Programming", Decision Sciences, Vol. 5 (1974), pp. 36-44.

[10] LAUGHHUM, D. L., "Quadratic Binary Programming with Application to Capital Budgeting Problems", Operations Research, Vol. 18, No. 3 (1970), pp. 454-461.

[11] SENJIU, S. and Y. TOYODA, "An Approach to Linear Programming with 0-1 Variables", Management Science, 15 (1968), pp. B196-207.

[12] TOYODA, Y., "A Simplified Algorithm for Obtaining Approximate Solutions to 0-1 Programming Problems", Management Science, 21 (1975), pp. 1417-1427.

[13] WATTERS, L. J., "Reduction of Integer Polynomial Programming Problems to Zero-One Linear Programming Problems", Operations Research, Vol. 15 (1967), pp. 1171-1174.

[14] ZANAKIS, S. M., "Heuristic 0-1 Linear Programming: An Experimental Comparison of Three Methods", Management Science, 24 (1977), pp. 91-104.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER #81	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) EFFICIENT HEURISTIC ALGORITHMS FOR POSITIVE 0-1 POLYNOMIAL PROGRAMMING PROBLEMS		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL REPORT
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Frieda Granot		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0418
9. PERFORMING ORGANIZATION NAME AND ADDRESS DEPARTMENT OF OPERATIONS RESEARCH STANFORD UNIVERSITY, STANFORD, CALIF.		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-061
11. CONTROLLING OFFICE NAME AND ADDRESS OPERATIONS RESEARCH PROGRAM CODE 434 OFFICE OF NAVAL RESEARCH ARLINGTON, VIRGINIA 22217		12. REPORT DATE August 1978
		13. NUMBER OF PAGES 24
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE AND SALE; DISTRIBUTION IS UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This report also issued as Technical Report SOL-#78-20 on Contract Contract F 44620-74-C-0079, Air Force.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Integer Polynomial Programming
Integer Programming
Heuristic Algorithms
Algorithms
Polynomial Programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE REVERSE SIDE

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

EFFICIENT HEURISTIC ALGORITHMS FOR POSITIVE 0-1 POLYNOMIAL PROGRAMMING PROBLEMS

by Frieda Granot

We consider in this paper the positive 0-1 polynomial programming (PP) problem of finding a 0-1 n-vector x that maximizes $c^T x$ subject to $f(x) \leq b$ where $c, b \geq 0$ and f is an m-vector of polynomials with non-negative coefficients.

Two types of heuristic methods for solving polynomial programming (PP) problems were developed. The various algorithms were tested on randomly generated problems of up to 1000 variables and 200 constraints. Their performance in terms of computational time and effectiveness was investiaged. The results were extremely encouraging. Optimal solutions were consistently obtained by some of the heuristic methods in over 50% of the problems solved. The effectiveness was on the average better than 99% and no less than 96.5%. The computational time using the heuristic for PP problems is on the average 5% of the time required to solve the problems to optimality.

78-20/81